MATRIX OPERATIONS ROM
11270B & OPTION 270
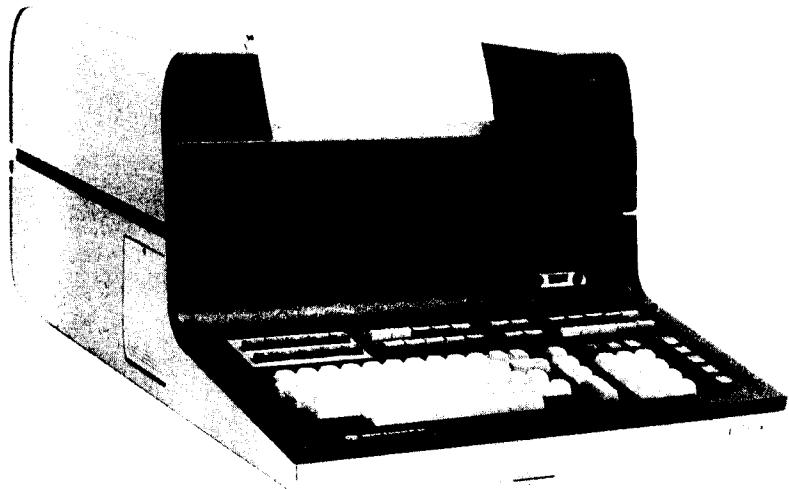
9830A CALCULATOR SHOWN WITH 9866A PRINTER

TABLE OF CONTENTS

◆━◆━◆━◆━◆━◆━◆━◆━◆━◆━◆━◆━◆

PREFACE

The Matrix Operations Read-Only-Memory (ROM) can be purchased as an accessory plug-in block or as an internal modification to the calculator.

The Plug-in Version:
>   The 11270B Matrix Operations ROM block is installable by the user. It plugs into any of the five slots behind the ROM door on the left side of the calculator.

The Calculator Modification:
>   The Option 270 Matrix Operations ROM must be installed by qualified HP personnel. When it is installed, a decal showing the option number (Option 270) is attached to the inside of the ROM door.

>   Should you wish to add the option after you have received your calculator, please order accessory number HP 11270F from the sales office nearest to you (see the inside back cover of this manual). The Option 270 will then be installed for you by our field personnel.

Once either version of the ROM (the plug-in block or the internal modification) has been installed, the operation is identical. Therefore, this manual makes no further distinction between the two types of ROM.

Chapter 1

INTRODUCTORY DESCRIPTION

The Matrix Operations Read-Only-Memory (the Matrix ROM) provides additional capabilities to the Model 30, permitting mathematical operations upon matrices, useful in physics, engineering and statistics operations. It also provides special programming techniques useful in processing any kind of data in tabular or array form.

◆◆◆◆◆ EQUIPMENT SUPPLIED ◆◆◆◆◆

One Operating Manual, -hp- Part Number 09830-90004, is supplied with the Matrix ROM.

◆◆◆◆ INSPECTION PROCEDURE ◆◆◆◆

Refer to Appendix A in the 9830A Calculator Operating and Programming Manual for the procedures used to verify the operation of ROM's.

◆◆◆ INSTALLING THE PLUG-IN BLOCK ◆◆◆

The complete procedure to install a plug-in block is in the Operating and Programming Manual for the 9830A Calculator. Following are some reminders:

The block can be installed in any of the five ROM slots.

Switch the calculator off before installing or removing a block.

The label on the block should be 'right-side-up' and facing the ROM door when the block is properly installed.

Ensure that the block is properly mated to the connector at the back of the slot before switching the calculator on.

◆◆◆◆ OTHER REQUIREMENTS ◆◆◆◆

It is assumed that you are already familiar with BASIC programming and with the operating procedures for the HP 9830A Calculator.

NOTES

Chapter 2

CHARACTERISTICS OF MATRICES

A table of data, or any collection of data elements arranged in rows and columns, is known as a matrix. A list of data, or any collection of data elements arranged in a single column or row, is known as a vector.

Here is an example of a matrix:

GRADE	BOYS	GIRLS
1	10	7
2	9	8
3	9	10
4	7	9
5	7	10
6	9	11

Here is an example of a vector:

TEST SCORES
93
85
79
89
68
95
100

◆◆◆◆◆ MATRIX NAMES ◆◆◆◆◆◆

A matrix name consists of any single letter from A through Z, hence, up to 26 matrices can be assigned at the same time in one program.

◆━◆━◆━◆━ MATRIX DEFINITION ━◆━◆━◆━◆

To reserve storage space for arrays (matrices and vectors), a DIM or COM* statement is used. Arrays not mentioned in a DIM statement are assumed to have 10 elements if they are one-dimensional, or 10 rows and 10 columns if they are two-dimensional.

The number of elements in a vector, or the number of rows or columns in a matrix, must be specified as an integer from 1 to 256 in a DIM statement as:

```
10 DIM A[100]
20 DIM B[75,5]
30 DIM C[20,20]
```

In statement 10 above, Array A is a column vector of 100 row elements.

In statement 20, Array B is dimensioned 75 rows by 5 columns. This row-column convention exists throughout this manual.

In statement 30, Array C has the same number of rows and columns, and is square. A square matrix is any matrix having the same number of rows and columns. Some matrix operations, such as inversion, can be performed only on square matrices.

Generally, the operations which apply to matrices also apply to vectors. So it is convenient to think of a column vector of n elements as an 'n by 1' matrix, and a row vector of n elements as a '1 by n' matrix.

In this manual, arrays, whether they are one-dimensional or two-dimensional, are referred to as matrices. A distinction is made only when special rules apply to the use of vectors.

Space can be saved when *storing* large quantities of array data by specifying integer or split precision. This is because while full (normal) precision requires four words of calculator memory per array element, split precision requires two words and integer precision requires only one word per array element. Split and integer precision are specified in the DIM statement as shown below.

```
10 DIM AS[200,5]
20 DIM BI[100,14]
```

Although calculator memory space can be saved when storing data by using integer and split precision, the determinant and inverse of a matrix should be performed only on full precision matrices. This is because, when integer and split precision are specified, errors due to reiterative internal rounding are multiplied during complex operations and often contribute to illogical results. Therefore, specify full precision matrices whenever possible, split precision with the greatest care, and avoid integer precision when inverting or finding the determinant of a matrix.

---

* The COM statement can also be used to reserve storage space for arrays, but must be the first statement entered into the calculator memory. Refer to the Model 30 Operating and Programming Manual for more information about the COM statement.

◆━◆━◆━◆ MATRIX BOUNDARIES ◆━◆━◆━◆

The working size of a matrix can be smaller than its physical size, which is the space reserved in the DIM statement. For example, a matrix specified 20 by 20 in a DIM statement can be used to store fewer than 400 data elements; the DIM statement supplies only an upper bound on the number of elements.

NOTES

Chapter 3

INPUT AND OUTPUT OF MATRICES

This chapter describes some of the methods for storing a table or a list of data in a matrix and for printing a table or list of data from a matrix.

◆━━◆━━◆━━◆ THE INPUT STATEMENT ◆━━◆━━◆━━◆

The INPUT statement, with the FOR and NEXT statements, makes it easy to enter a matrix from the keyboard. The following instructions enable a 5 by 3 matrix to be entered one row at a time.

```
10 DIM A[5,3]
20 FOR J=1 TO 5
30 INPUT A[J,1],A[J,2],A[J,3]
40 NEXT J
```

◆━━◆━━◆━━◆ THE READ STATEMENT ◆━━◆━━◆━━◆

You can include the values to be stored in a matrix in program DATA statements, and can use the READ statement, with the FOR and NEXT statements, to read the matrix. With the following instructions, a 3 by 2 matrix is read.

```
10 DIM A[3,2]
20 DATA 12,10,5,3,14,7
30 FOR M=1 TO 3
40 FOR N=1 TO 2
50 READ A[M,N]
60 NEXT N
70 NEXT M
```

◆━━◆━━◆━━◆ THE PRINT STATEMENT ◆━━◆━━◆━━◆

The PRINT statement provides control in printing of headings and other information with a matrix. Using the following instructions, the data in the 3 by 2 Matrix A, above, can be printed and labeled.

```
80 PRINT "GRADE  BOYS  GIRLS"
90 FOR M=1 TO 3
100 PRINT M;A[M,1];A[M,2]
110 NEXT M
```

◆◆◆◆ THE MAT READ STATEMENT ◆◆◆◆

Syntax:

MAT READ matrix name [(expression[,expression])] . . .

Although matrix values can be assigned individually by using the READ statement, the MAT READ statement is easier to use because it causes an entire matrix to be read. To read a 3 by 2 matrix with the MAT READ statement:

```
10 DATA 12,10,5,3,14,7
20 MAT READ A[3,2]
```

The MAT READ statement causes the matrix to be filled from the DATA statement in the conventional row-column order: 1,1; 1,2; 2,1; and so forth.

```
10 MAT READ A
20 MAT READ X[6]
30 MAT READ M[N,P]
40 MAT READ C,D,E[6,N]
```

A new working size (an implicit REDIM statement — see Chapter 4) may be specified, as in statements 20 and 30 above, but must be within the limits of the DIM statement.

If no size specification is included, as in 10, above, the working size of the matrix is assumed to be its current physical size. If your program does not contain a DIM specification and the working size is not specified in the MAT READ statement, a 10 by 10 matrix is assumed and the first 100 elements in the DATA statement are used, or a 10 element vector is assumed and the first 10 elements in the DATA statement are used. For example, the statement A(5) = 32 causes a vector to be assumed.

More than one matrix name can appear in a MAT READ statement, as in statement 40, above.

◆◆◆◆ THE MAT PRINT STATEMENT ◆◆◆◆

Syntax:

MAT PRINT matrix name [,matrix name] . . .

Although a matrix can be printed element-by-element by using a PRINT statement, the MAT PRINT statement is easier to use. The MAT PRINT statement causes an entire matrix to be printed row by row. Each row starts a new line, but when all the elements in a row will not fit in one line, the elements overflow into additional lines. Each row is separated by a blank line.

```
10 MAT PRINT A
20 MAT PRINT M;
30 MAT PRINT X,Y;
40 MAT PRINT A,X;Y
```

The spacing between row elements is controlled by the use of the comma or the semicolon. The comma causes the matrix to be printed with five elements per line, while the semicolon causes the matrix to be printed with up to twelve elements per line. The punctuation after the matrix name determines its spacing on a page. If there is no punctuation the matrix is printed with five elements per line.

A one-dimensional array, or column vector, is printed one element per line with double spacing between lines.

NOTES

Chapter 4

MATRIX OPERATIONS

The operations discussed in this chapter are designed specifically for use with matrices. However, all BASIC language operations that can be performed on simple variables can also be performed on array variables or matrix elements; such as IF, WRITE, SQR, ABS, etc.

◆◆◆◆ ADDITION OF MATRICES ◆◆◆◆

Syntax:

MAT matrix name = matrix name + matrix name

The corresponding elements of two matrices can be added by using a MAT (Matrix) command:

```
110 MAT C=A+B
```

The actual dimensions of A,B and C must be the same. Each element in A is added to the corresponding element in B and the result is stored in the corresponding position in C.

Calculator memory space can be saved if the result is accumulated in one of the matrices already containing data, such as Matrix X in:

```
120 MAT X=X+Y
```

Each element in Y is added to the corresponding element in X and the result is contained in X.

If Matrix A is a 2 by 3 matrix containing:

$$
\begin{array}{ccc}
3 & 6 & 2 \\
4 & 1 & 5
\end{array}
$$

and Matrix B is a 2 by 3 matrix containing:

$$
\begin{array}{ccc}
2 & 4 & 7 \\
5 & 3 & 1
\end{array}
$$

Then, if the statement MAT C=A+B is executed, Matrix C contains:

$$
\begin{array}{ccc}
5 & 10 & 9 \\
9 & 4 & 6
\end{array}
$$

◆━◆━◆━◆ SUBTRACTION OF MATRICES ◆━◆━◆━◆

Syntax:

MAT matrix name = matrix name − matrix name

The corresponding elements of two matrices can be subtracted:

```
110 MAT C=A-B
120 MAT X=X-Y
```

As in the addition of matrices, all three matrices must have the same actual dimensions.

◆━◆ SCALAR MULTIPLICATION OF MATRICES ◆━◆

Syntax:

MAT matrix name = (expression) * matrix name

Each element in a matrix can be multiplied by a number or by the value of any arithmetic expression. The number or expression must be enclosed in parentheses and must precede the matrix name:

```
110 MAT A=(5)*A
120 MAT X=(N/3)*Y
130 MAT P=(SQR7+3↑N)*R
```

Each matrix must have the same actual dimensions if more than one matrix is named. In statements 120 and 130, above, N is a simple variable.

◆━◆━◆━◆ COPYING MATRICES ◆━◆━◆━◆

Syntax:

MAT matrix name = matrix name

For an exact copy in Matrix A from Matrix B:

```
110 MAT A=B
```

Each matrix must have the same actual dimensions.

◆◆◆◆◆ EXAMPLE 1 ◆◆◆◆◆◆

Below are two tables containing the Math, Science, and Reading grades achieved by five students during two quarters of one school year.

	First Quarter				Second Quarter		
Student No.	Math	Science	Reading	Student No.	Math	Science	Reading
1	80	85	78	1	78	81	80
2	71	80	72	2	73	82	88
3	97	92	83	3	93	90	85
4	77	82	98	4	81	88	94
5	93	94	98	5	91	90	84

We will use the grades in the first quarter for Matrix A, and in the second quarter for Matrix B:

Matrix A:				Matrix B:		
80	85	78		78	81	80
71	80	72		73	82	88
97	92	83		93	90	85
77	82	98		81	88	94
93	94	98		91	90	84

If the statement MAT C=A+B is executed, Matrix C contains:

158	166	158
144	162	160
190	182	168
158	170	192
184	184	182

The grades for each student in each class have been added. We can now find the average of the two term grades for each student in each class:

```
110 MAT C=(1/2)*C
```

An expression and a scalar multiplication are used here; the result in C is now:

79	83	79
72	81	80
95	91	84
79	85	96
92	92	91

◆━━◆━━◆━━◆━━◆ **EXAMPLE 1** ◆━━◆━━◆━━◆━━◆
(Continued)

The matrix now represents the average grade for each student in each of the three subjects:

Average Mid-Year Grades

Student No.	Math	Science	Reading
1	79	83	79
2	72	81	80
3	95	91	84
4	79	85	96
5	92	92	91

NOTE

Only one operation is performed in a MAT command. You cannot say MAT A = (1/2)*B+C, but must perform two separate steps:

MAT A=(1/2)*B

MAT A=A+C

◆━━◆━━◆━━◆ **MATRIX MULTIPLICATION** ◆━━◆━━◆━━◆

Syntax:

MAT matrix name = matrix name * matrix name

Matrices can be multiplied only when the number of columns in the first matrix equals the number of rows in the second matrix.

```
110 MAT C=A*B
```

The matrix to the left of the equals sign must not appear to the right of the equals sign, (i.e., you cannot say MAT X=X*Y).

Unless you have already been introduced to matrix multiplication, you might assume that standard arithmetic rules are followed; however, this is not the case. In matrix multiplication, for MAT C=A*B, the elements in each column of Matrix B are multiplied by the corresponding elements in each row of Matrix A. The row products are then added together and stored in the appropriate row and column of Matrix C. Following are several examples of matrix multiplication.

Below is a table of ticket sales for four bus routes, and a table of ticket prices for the three kinds of tickets. Matrix multiplication can give you the total sales for each route.

Table A. Ticket Sales by Route

Route	Single Trip	Round Trip	Commuter
1	143	200	18
2	49	97	24
3	314	77	22
4	82	65	16

Table B. Ticket Prices

Single Trip	.25
Round Trip	.45
Commuter	18.00

Here are the instructions to read the values in Tables A and B into Arrays A and B, respectively, and perform the matrix multiplication (MAT C=A*B).

```
10 DIM A[4,3],B[3],C[4]
20 DATA 143,200,18,49,97,24,314,77,22,82,65,16
30 DATA 0.25,0.45,18
40 MAT READ A[4,3],B[3]
50 MAT C=A*B
55 FIXED 2
60 MAT PRINT C
```

After A and B have been multiplied, Matrix C contains the total sales, in dollars, for each route: C(1) contains sales for Route 1, C(2) contains sales for Route 2; and so forth.

Matrix C:	449.75	(.25*143+.45*200+18*18)
	487.90	(.25*49+.45*97+18*24)
	509.15	(.25*314+.45*77+18*22)
	337.75	(.25*82+.45*65+18*16)

In the above example, a 4 by 3 matrix, Matrix A, is multiplied by a 3 by 1 matrix, Matrix B, giving a 4 by 1 matrix, Matrix C.

Suppose a price change is being considered, and Matrix B contains two columns of ticket prices:

	Old Price	New Price
Single Trip	.25	.30
Round Trip	.45	.50
Commuter	18.00	17.00

Then Matrix A, a 4 by 3 matrix, multiplied by Matrix B, a 3 by 2 matrix, results in Matrix C, a 4 by 2 matrix:

449.75	448.90	(.30*143+.50*200+17.00*18)
487.90	471.20	(.30*49+.50*97+17.00*24)
509.15	506.70	(.30*314+.50*77+17.00*22)
337.75	329.10	(.30*82+.50*65+17.00*16)

Here are the instructions to perform this multiplication. Note that the DATA in statement 30 has been rearranged.

```
10 DIM A[4,3],B[3,2],C[4,2]
20 DATA 143,200,18,49,97,24,314,77,22,82,65,16
30 DATA 0.25,0.3,0.45,0.5,18,17
40 MAT READ A[4,3],B[3,2]
50 MAT C=A*B
55 FIXED 2
60 MAT PRINT C
```

MATRIX MULTIPLICATION
(Continued)

Mathematically, the product of two matrices, MAT C=A*B, is represented as follows:

$$C(I,K)=\sum_{J=1}^{N} A(I,J)*B(J,K)$$

Where N is the number of columns in Matrix A, and rows in Matrix B.

For any matrix multiplication, MAT C=A*B, if the dimensions of A=(P,N) and B=(N,Q), the result is a matrix of dimensions (P,Q). For example, a 5 by 4 matrix multiplied by a 4 by 1 matrix results in a 5 by 1 matrix. Remember that the value of N, above, must be the same in the two matrices: you cannot multiply a 5 by 3 matrix by a 5 by 3 matrix, but you can multiply a 5 by 3 matrix by a 3 by 5 matrix, or you can multiply a 3 by 5 matrix by a 5 by 3 matrix. However, the result of MAT C=B*A is not necessarily the same as MAT C=A*B.

TRANSPOSITION OF MATRICES

Syntax:

MAT matrix name = TRN (matrix name)

The transposition of a matrix causes the rows in the matrix to become columns, and the columns to become rows:

```
110 MAT A=TRN(B)
```

The same matrix cannot appear on both sides of the equals sign. The actual dimensions of the transposed matrix must equal the reverse of the dimensions of the original matrix, that is, if the dimensions of B=(P,Q), transposition of Matrix B results in a matrix of dimensions (Q,P).

The transposition of:

2	3
4	5
6	7

is:

2	4	6
3	5	7

Row 1 in the first matrix becomes column 1 in the second. Then row 2 becomes column 2 and so on.

Below are some examples which make use of matrix transposition.

Suppose you have a table of quantities of three product lines sold at four locations, and a second list consisting of prices for each product line.

Table A. Product Sales by Location			
Location	Prod 1	Prod 2	Prod 3
1	7	2	6
2	3	8	9
3	10	5	4
4	3	1	2

Table B. Product Prices	
Product	Price
1	5
2	2
3	1

Table A is read into Matrix A, and Table B is read into Matrix B with the following instructions:

```
10 DIM A[10,10],B[10,10],C[4,1]
20 MAT READ A[4,3],B[3,1]
30 DATA 7,2,6,3,8,9,10,5,4,3,1,2,5,2,1
```

Now the two matrices can be multiplied to determine the total sales for each location:

```
40 MAT C=A*B
```

Matrix C is now a column vector containing 4 elements. C(1,1) is the sales at location 1, C(2,1) is the sales at location 2, and so forth.

To obtain a printout of sales by product at each location, and the total sales at each location, both of these matrices must be transposed:

```
50 DIM D[3,4],E[1,4]
60 MAT D=TRN(A)
70 MAT E=TRN(C)
```

Now headings and data can be printed:

```
80 PRINT "LOCATION 1   LOCATION 2   LOCATION 3   LOCATION 4"
90 MAT PRINT D
100 PRINT "TOTALS"
110 MAT PRINT E
```

◆◆◆◆◆ **EXAMPLE 2** ◆◆◆◆◆

A magazine publishing company has a table of distribution for magazines by region.

Table A. Magazine Distribution by Region

Region	Magazine A	Magazine B
1	200	50
2	150	45
3	250	60

A table of magazine cost-per-copy is used to determine the amount invested in each region:

Table B. Cost-per-copy by Magazine

Magazine	Cost-per-copy
A	.32
B	.18

Here is a program used to read the data and perform matrix multiplication to determine magazine cost by region.

```
10 DIM A[3,3],B[2,1],C[3]
20 MAT READ A[3,2],B
30 DATA 200,50,150,45,250,60,0.32,0.18
40 MAT C=A*B
50 FOR I=1 TO 3
55 FORMAT "REGION",F2.0,"    COST",F6.2
60 WRITE (15,55)I,C[I]
70 NEXT I
```

Here is the result:

```
REGION 1    COST 73.00
REGION 2    COST 56.10
REGION 3    COST 90.80
```

The data contained in Matrix A is now to be used to determine the number of complimentary copies to be allotted to each region. A table of percentages by region is used:

Table D: % Complimentary Distribution by Region

Region	% Complimentary
1	15
2	13
3	21

The data from table D is contained in Matrix D. To perform a matrix multiplication, the columns in the first matrix must match the rows in the second matrix, (i.e., both must represent 'region'). To meet this requirement, Matrix A is transposed and stored in Matrix E. Here are additional program steps for this second matrix multiplication.

```
90 DIM D[3,1],E[2,3],F[2]
100 DATA 0.15,0.13,0.21
110 MAT READ D
120 MAT E=TRN(A)
130 MAT F=E*D
140 PRINT "MAGAZINE A COMPLIMENTARIES";INT(F[1])
150 PRINT "MAGAZINE B COMPLIMENTARIES";INT(F[2])
```

Here is the result:

```
MAGAZINE A COMPLIMENTARIES 102
MAGAZINE B COMPLIMENTARIES 25
```

◆◆◆◆ THE CONSTANT MATRIX ◆◆◆◆

Syntax:

MAT matrix name = CON [(expression[, expression])]

A constant matrix is any matrix with all elements equal to 1.

The statement MAT B=CON(3,3) results in the following 3 by 3 matrix:

$$
\begin{matrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{matrix}
$$

Since 1 has a logical value of 'true', the constant matrix is useful for logic initialization.

If no subscripts are given, the constant matrix has the most recent dimensions specified. If subscripts are given, they cause the array to be dimensioned to the specified size.

One example of the use of a constant matrix, in summarizing a table of data, follows on page 4-12.

◆◆◆ THE REDIM STATEMENT ◆◆◆

Syntax:

REDIM (expression[, expression])

The working size of a matrix is the same as the physical size unless otherwise specified. To change the working size, a size specification is included in a MAT READ, CON, ZER, IDN or REDIM statement.

NOTE

When a new working size is specified for a matrix containing data, the data is rearranged.

The following instructions create Matrix A:

```
10 DIM A[20,20]
20 MAT READ A[2,3]
30 DATA 4,7,9,5,8,10
40 MAT PRINT A;
```

Matrix A:

```
4       7       9

5       8       10
```

Matrix A has a working size of 2 by 3. The following REDIM statement changes the working size to 3 by 2.

```
50 REDIM A[3,2]
60 MAT PRINT A;
```

Matrix A is now:

```
4       7

9       5

8       10
```

When you have a large program, a matrix which is no longer used can be redimensioned, thus using the same space in the calculator a second time.

Matrix A is a 2 by 4 matrix containing student absences for the school year as follows:

	First Quarter	Second Quarter	Third Quarter	Fourth Quarter
Boys	62	85	73	81
Girls	70	87	65	83

The average of absences for boys and girls for the school year is calculated:

```
10 DIM C[2,1]
20 MAT READ A[2,4]
25 FIXED 0
30 DATA 62,85,73,81,70,87,65,83
40 MAT B=CON[4,1]
50 MAT C=A*B
60 MAT C=(1/4)*C
70 PRINT "AVERAGE ABSENCES"
80 PRINT "  BOYS      GIRLS"
90 PRINT C[1,1];C[2,1]
```

Matrix A is now to contain total enrollment by month, and average enrollment is to be calculated:

```
100 REDIM A[1,9]
110 FOR M=1 TO 9
120 DISP "ENTER MONTHLY ENROLLMENT";
130 INPUT A[1,M]
140 NEXT M
150 REDIM B[9,1],C[1,1]
160 MAT B=CON[9,1]
170 MAT C=A*B
180 MAT C=(1/9)*C
190 PRINT "AVERAGE ENROLLMENT"
200 MAT PRINT C
```

Another use of the REDIM statement is shown later in Example 4, lines 40 to 90, where a matrix initially contains two rows of elements supplied by a MAT READ statement. After the REDIM statement is used, the matrix contains four rows of elements; and the values of the elements in the third and fourth rows of the matrix are obtained from calculations.

◆━━◆━━◆━━◆━━◆ **EXAMPLE 3** ◆━━◆━━◆━━◆━━◆

Suppose you have a table of inventory values by stock number at five branch stores, as in Table A, below.

Table A. Inventory Value

Stock Number	Branch A	Branch B	Branch C	Branch D	Branch E
1	52.80	35.20	31.68	26.40	26.40
2	37.50	31.25	25.00	23.75	18.75
3	8.25	6.60	6.60	4.95	3.30
4	11.76	10.50	8.40	8.40	6.30

The following instructions are used to establish the 4 by 5 matrix, Matrix A, then to calculate and print the total inventory value for each stock number.

```
10 DIM A[4,5],B[5,1],C[5,1]
20 DATA 52.8,35.2,31.68,26.4,26.4
30 DATA 37.5,31.25,25,23.75,18.75
40 DATA 8.25,6.6,6.6,4.95,3.3
50 DATA 11.76,10.5,8.4,8.4,6.3
60 MAT READ A[4,5]
70 MAT B=CON[5,1]
80 REDIM C[4,1]
90 MAT C=A*B
100 PRINT "TOTAL INVENTORY VALUE BY STOCK NUMBER"
110 FOR I=1 TO 4
120 FORMAT F10.0,F8.2,/
130 WRITE (15,120)I,C[I,1]
140 NEXT I
```

Matrix A is multiplied by the constant matrix, Matrix B, to give the total value across each row in A: 1*52.80+1*35.20+1*31.68, and so forth.

The following instructions use the above Matrix A, and calculate and print the total inventory value for each branch store.

```
150 DIM D[5,4],E[1,5]
160 REDIM C[5,1]
170 MAT D=TRN(A)
180 MAT B=CON[4,1]
190 MAT C=D*B
195 FIXED 2
200 PRINT "TOTAL INVENTORY VALUE BY STORE"
210 PRINT "STORE A        STORE B        STORE C";
215 PRINT "        STORE D        STORE E"
220 MAT E=TRN(C)
230 MAT PRINT E
```

In these instructions, Matrix C is transposed before it is printed. The stores can then be used as column headings so that the results are easily identified.

◆━━◆━━◆━━◆ THE ZERO MATRIX ◆━━◆━━◆━━◆

Syntax:

MAT matrix name = ZER [(expression[,expression])]

A zero matrix is any matrix with all elements equal to zero.

The statement MAT B=ZER(3,3) results in the following 3 by 3 array:

```
            0   0   0
            0   0   0
            0   0   0
```

Since 0 has a logical value of 'false', the zero matrix is useful for logic initialization.

If no subscripts are given, the zero matrix has the most recent dimensions specified. If subscripts are given, they cause the array to be dimensioned to the specified size.

Here is an example of the use of the zero matrix. Table A contains the grades received in Reading by 20 students. Below is a program to read the grades and print out the number of students receiving each grade.

Table A. Student Grades in Reading

Student	Grade	Student	Grade
1	93	11	52
2	85	12	66
3	79	13	80
4	89	14	79
5	68	15	98
6	95	16	95
7	100	17	89
8	66	18	68
9	79	19	72
10	85	20	96

```
10 DATA 93,85,79,89,68,95,100,66,79,85
20 DATA 52,66,80,79,98,95,89,68,72,96
30 DIM A[100]
40 MAT A=ZER
50 FOR I=1 TO 20
60 READ B
70 A[B]=A[B]+1
80 NEXT I
90 PRINT "GRADE,  NUMBER OF STUDENTS"
100 FOR I=1 TO 100
110 IF A[I]=0 THEN 130
120 PRINT I,A[I]
130 NEXT I
```

◆◆◆◆ **THE IDENTITY MATRIX** ◆◆◆◆

Syntax:

MAT matrix name = IDN [expression , expression]

An identity matrix is a square matrix containing zeros with a principal diagonal containing all ones.

The statement MAT B=IDN(3,3) results in the following 3 by 3 matrix:

$$\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}$$

If no subscripts are given, the identity matrix has the most recent dimensions specified. If subscripts are given, they must be equal, and they cause the array to be dimensioned to the specified size:

```
10 MAT A=IDN
20 MAT B=IDN[5,5]
30 MAT X=IDN[0,0]
```

The identity matrix is defined as the matrix which, when multiplied by any Matrix A results in Matrix A. It is used by mathematicians in fields such as numerical analysis in determining the accuracy of inversions.

◆◆◆◆ **INVERSION OF MATRICES** ◆◆◆◆

Syntax:

MAT matrix name = INV (matrix name)

The inverse of a Matrix A is a Matrix B which, when multiplied by the original Matrix A, produces the identity matrix. Both Matrix A and B should be full precision matrices to insure accuracy. Only a square matrix can be inverted:

```
110 MAT A=INV(B)
120 MAT X=INV(X)
```

The same matrix may appear on both sides of the equals sign, as in statement 120, above.

In performing the inversion, the calculator must generate an additional internal work area. For an N by N matrix, the additional amount of internal work area required is N words of calculator memory.

Inversions can be used to help solve sets of simultaneous linear equations. Here is a word problem which is solved by using simultaneous linear equations:

John bought 3 oranges and 4 apples for 47 cents, and Mary bought 2 oranges and 2 apples for 28 cents. How much is 1 orange? How much is 1 apple?

We set up the following equations from the problem:

$$3X + 4Y = 47$$
$$2X + 2Y = 28$$

Using the values of the coefficients on the left side of the equations in Matrix A, we have the following 2 by 2 matrix:

$$\begin{matrix} 3 & 4 \\ 2 & 2 \end{matrix}$$

Using the values of the constants on the right side of the equations in Matrix B, we have the following 2 by 1 matrix:

$$\begin{matrix} 47 \\ 28 \end{matrix}$$

To find the values for X and Y, we use the matrix inversion command and a matrix multiplication:

```
40 MAT A=INV(A)
50 MAT C=A*B
```

Now the price of an orange, X, is contained in C(1,1), and the price of an apple, Y, is contained in C(2,1).

Here are the program instructions to perform the matrix inversion and print the result.

```
10 DIM A[2,2],B[2,1],C[2,1]
20 MAT READ A,B
30 DATA 3,4,2,2,47,28
40 MAT A=INV(A)
50 MAT C=A*B
60 FORMAT F4.0," CENTS EACH",/,F4.0," CENTS EACH"
70 WRITE (15,60)"APPLES ",C[1,1],"ORANGES",C[2,1]
```

Even though a matrix is square, it may not have an inverse. A calculator error will occur when an inversion is attempted and the matrix has no inverse. Therefore, it is sometimes useful to insure that the matrix has an inverse by making sure that its determinant is not equal to zero. More discussion of the determinant follows.

For a more detailed explanation of matrix inversion, you can refer to an advanced mathematics textbook.

◆━━◆━━◆━━◆━━◆ DETERMINANTS ◆━━◆━━◆━━◆━━◆

Syntax:

> DET (matrix name)

The DET function calculates the determinant of a square matrix:

```
10 X=DET(A)
20 Y=X+DET(B)
```

A determinant is a measure of the independence of the rows (or columns) of a matrix. The DET function returns a numerical value to an expression. For an N by N matrix, the additional amount of internal work area required is equal to the original size of the matrix plus N words. The matrix for which a determinant is evaluated should be full precision to insure accuracy.

If the determinant of a square matrix of simultaneous linear equations is not zero, the system of equations has precisely one solution; that is, the inverse of the matrix can be obtained. On the other hand, if the determinant is zero, the system of equations has no precise solution, and the matrix has no inverse. Therefore, it may be useful to use the DET function to examine the matrix before inversion is performed in order to prevent a calculator error condition. Following is an example:

```
10 DIM A[2,2],B[2,2]
20 DATA 4,3,5,1
30 MAT READ A
40 IF DET(A)=0 THEN 70
50 MAT B=INV(A)
60 GOTO 80
70 MAT B=A
80 MAT PRINT A;B;
90 PRINT "DET A",DET(A)
```

◆━━◆━━◆━━◆━━◆ EXAMPLE 4 ◆━━◆━━◆━━◆━━◆

A linear relationship exists between the daily temperature and a vending machine company's sale of soft drinks. The company wishes to be able to predict sales based on daily temperatures. This example shows how the company might do this by performing a regression analysis using the statistical method known as least squares fit.

Table 4-1 contains some of the sales figures and daily maximum temperatures.

Table 4-1. Sales Figures vs. Maximum Daily Temperatures

Maximum Temperature	Sales ($)
81	41
64	30
57	26
76	36
92	49

The company performs a simple linear regression to find the relationship between temperatures and sales. If this is done by hand, first a computation table like Table 4-2 is needed, where X values are temperatures, and Y values are sales.

Table 4-2. Computation Table for Regression

X	Y	X*Y	X^2
81	41	3321	6561
64	30	1920	4096
57	26	1482	3249
76	36	2736	5776
92	49	4508	8464
370	182	13967	28146

The value of sales (Y) can be predicted for any temperature (X) by using the formula Y=A+BX. But first the values of A and B must be calculated. To find A and B, the following two simultaneous equations are solved:

$$An+B\Sigma X=\Sigma Y$$
$$A\Sigma X+B\Sigma X^2 =\Sigma XY$$

Where n is the number of entries, in this case five.*

From the computation table, we know the following values:
$$\Sigma X=370; \Sigma Y=182; \Sigma XY=13967 \text{ and } \Sigma X^2 =28146.$$

Now the two equations are

$$\text{I } 5A+370B=182$$
$$\text{II } 370A+28146B=13967$$

To solve the simultaneous equations by hand, we eliminate A from the equations by multiplying the first equation by 370 and the second by 5 and then subtracting Equation I from Equation II.

$$\text{II } 1850A+140730B=69835$$
$$\text{I } 1850A+136900B=67340$$
$$\overline{3830B=2495}$$

$$B=2495/3830B=.6514$$

Although it would be easy by the method of substitution to find the value of A, now that the value of B is known, the value of A is of little significance in this case. However, the regression coefficient, B, is of interest: it represents an estimate of the average increase in sales for each additional degree of temperature. That is, within the temperatures studied, each additional degree in temperature increases sales by an average of 0.65 dollars.

The above procedure is repeated in the program steps which follow. First, Matrix A is set up, containing the X, Y, XY, and X^2 values shown in the computation table. The summation (Σ) values are then calculated and stored in Matrix C: C(1,1)=ΣX; C(2,1)=ΣY; C(3,1)=ΣXY; and C(4,1)=ΣX^2.

---

* Increasing the number of entries in the computation table results in a more accurate prediction; however, for purposes of explanation, five entries are sufficient.

◆◆◆◆◆◆ **EXAMPLE 4** ◆◆◆◆◆◆
(Continued)

```
10 DATA 81,64,57,76,92
20 DATA 41,30,26,36,49
30 DIM A[4,5],B[5,1],C[4,1]
40 MAT READ A[2,5]
50 REDIM A[4,5]
60 FOR I=1 TO 5
70 A[3,I]=A[1,I]*A[2,I]
80 A[4,I]=A[1,I]*A[1,I]
90 NEXT I
100 MAT B=CON
110 MAT C=A*B
120 MAT PRINT A,C
```

To solve for A and B, we must perform a matrix inversion on the coefficients of the following simultaneous equations:

$$5A+C(1,1)B=C(2,1)$$
$$C(1,1)A+C(4,1)B=C(3,1)$$

The coefficients on the left of the equals sign are contained in Matrix M, and those on the right are in Matrix N.

Matrix M	5	C(1,1)	Matrix N	C(2,1)
	C(1,1)	C(4,1)		C(3,1)

The following instructions create Matrix M and Matrix N, and perform the necessary inversion and multiplication so that Matrix P contains the value of A, in P(1,1), and B, in P(2,1).

```
130 DIM M[2,2],N[2,1],P[2,1]
140 M[1,1]=5
150 M[1,2]=C[1,1]
160 M[2,1]=C[1,1]
170 M[2,2]=C[4,1]
180 N[1,1]=C[2,1]
190 N[2,1]=C[3,1]
200 MAT M=INV(M)
210 MAT P=M*N
215 FIXED 2
220 PRINT "THE REGRESSION COEFFICIENT IS ";P[2,1]
```

APPENDIX

◆━◆━◆━━◆ **DIAGNOSTIC NOTES** ◆━◆━◆━◆

INDICATION	MEANING
ERROR 66	Matrix must be square for attempted operation.
ERROR 67	New dimensions exceed existing DIM specifications.
ERROR 68	Matrix has no inverse. The data contained in the matrix does not have a solution.
ERROR 69	Incompatible dimensions. Dimensions of added, subtracted, multiplied, copied or transposed matrices must agree.

◆◆◆◆ MATRIX OPERATIONS ◆◆◆◆

DIM	10 DIM A[20,30],B[10,5]	Reserves space for a matrix of the specified physical dimensions. Initially, the working size is the same as the physical size for a matrix, as specified in the DIM statement.
REDIM	20 REDIM A[10,15]	Specifies a new working size for a matrix. The working size must not be greater than the physical size.
MAT READ	30 MAT READ A 31 MAT READ X,Y[5,5]	Reads an entire matrix from DATA statements. The matrix is filled in conventional row-column order. More than one matrix can be included in a MAT READ statement, and a new working size can be specified.
MAT PRINT	40 MAT PRINT A 41 MAT PRINT Y,Y;Z;	Prints an entire matrix row by row, with spacing of the columns controlled by the use of the comma and the semicolon. More than one matrix can appear in a MAT PRINT statement.
MAT +	50 MAT C=A+B 51 MAT X=X+Y	Matrix addition. The same matrix can appear on both sides of the equals sign.
MAT −	60 MAT C=A−B 61 MAT X=X−Y	Matrix subtraction. The same matrix can appear on both sides of the equals sign.
MAT =	70 MAT A=B	Copies values of Matrix B into Matrix A.
SCALAR *	80 MAT A=(1/5)*B 81 MAT B=(2*P1)*B	Each element in Matrix B is multiplied by the scalar value (expression in parentheses). The same matrix can appear on both sides of the equals sign.
MAT *	90 MAT A=B*C	Matrix multiplication. If the dimensions of A=(P,N) and B=(N,Q), the resulting matrix has dimensions (P,Q). The same matrix cannot appear on both sides of the equals sign.
MAT TRN	100 MAT A=TRN(B)	Transposes a matrix. If the dimensions of B=(P,N), the resulting matrix has dimensions (N,P). The same matrix cannot appear on both sides of the equals sign.
MAT CON	110 MAT A=CON 111 MAT B=CON[10,15]	Sets all elements of the matrix equal to 1. A new working size can be specified.
MAT ZER	120 MAT B=ZER 121 MAT X=ZER[5,5]	Sets all elements of the matrix equal to 0. A new working size can be specified.
MAT IDN	130 MAT C=IDN 131 MAT X=IDN[4,4]	Establishes an identity matrix — a square matrix containing all zeros with a principal diagonal of ones.
MAT INV	140 MAT A=INV(B) 141 MAT X=INV(X)	Inverts a square matrix. A matrix can be inverted into itself.
DET	150 IF DET(A)=0 THEN 180 151 X=DET(A)+DET(B)	Returns the value of the determinant of a square matrix to an expression.